Simo Pitkänen

Äänen tekninen toteutus mobiilipelissä

Metropolia Ammattikorkeakoulu

Medianomi (AMK)

Elokuvan ja television ko.

Opinnäytetyö

4.5.2015

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Simo Pitkänen Äänen tekninen toteutus mobiilipelissä 21 sivua + 3 liitettä 4.5.2015
Tutkinto	Medianomi (AMK)
Koulutusohjelma	Elokuvan ja television ko.
Suuntautumisvaihtoehto	Ääni
Ohjaaja(t)	lehtori Päivi Takala

Tämän opinnäytteen keskeisenä tutkimuskysymyksenä perehdytään mobiilipelin äänen tekniseen toteutusprosessiin. Tutkimuskysymystä käsitellään iOS-alustalle kehitetyn tapausesimerkin kautta. Opinnäytteen tarkoituksena on selvittää käytännön kautta, millaisia työvaiheita mobiilialustalle kehitetyn pelin äänisuunnitteluprosessi sisältää, ja mitä eri tekijöitä tämän prosessin aikana tulee ottaa huomioon.

Kirjoittaja avaa pelien historiaa menneisyydestä nykypäivään, ja ottaa esille mobiiliteknologian kehityksen vaikutuksen pelaamisen suosion kasvuun. Mobiilipeleille määritellään ominaisimmat piirteet suhteessa pc- ja konsolipeleihin. Opinnäytteessä taustoitetaan tapausesimerkkiin nojaten äänituotannon vaiheet esituotannosta implementointiin asti, ja tuodaan esille peliäänen funktioihin ja kategorioihin liittyviä huomioita. Kirjoittaja valottaa myös omaa taustaansa äänisuunnittelun opiskelijana, jolla ei ole aiempaa ekstensiivistä kokemusta peliäänisuunnittelusta.

Opinnäyte pureutuu mobiilialustojen kehitystyössä huomioon otettaviin teknisiin rajoitteisiin, jotka liittyvät laitekohtaisiin spesifikaatioihin ja spesifikaatioiden valmistaja- ja mallikohtaiseen varianssiin. Kirjoittaja pohtii näiden teknisten rajoitteiden vaikutusta suunnittelutyöhön. Tutkimuksessa käsitellään myös peliäänen teknistä toteutusta varten kehitettyjä väliohjelmistoja ja niiden tarjoamia ratkaisuvaihtoehtoja. Esille otetaan myös ohjelmointiin liittyvää peruskäsitteistöä, ja äänen implementointityössä käytettäviä ohjelmointikeinoja. Kirjoittaja esittelee peliä varten suunnittelemansa yksinkertaisen adaptiivisen musiikin järjestelmän.

Opinnäytteen teososana toimii SongHi Entertainment Oy -nimisen yrityksen kehittämä ja kirjoittajan äänisuunnittelema match 3-pelimekaniikkaan perustuva mobiilipeli ”Diamond Duo”.

Avainsanat	äänisuunnittelu, peliääni, väliohjelmisto, adaptiivinen musiikki

Author Title Number of Pages Date	Simo Pitkänen The technical process of mobile game audio design 21 pages + 3 appendices 4 May 2015
Degree	Bachelor of Arts
Degree Programme	Film and Television
Specialisation option	Sound Design and Production
Supervisor	Päivi Takala, Principal Lecturer of Sound

This final project focuses on the technical process of mobile game audio design. It reports on the development of the game Diamond Duo of SongHi Entertainment Ltd. In the project, the author was responsible for the audio design. The goal of the study is to define the individual production phases from the audio technical point of view, and to find out what kind of factors need to be taken into account in the process.

The author gives an overview of the history of video games and discusses the effect the advances in mobile technology have had on the rising popularity of gaming. Typical features of mobile games are contrasted to PC and console games. Diamond Duo is used as an example to shed light on the different phases of game audio production. The author introduces the general functions and categories of game audio. The report is written from the point of view of a sound design student who has little previous experience in game audio design.

The study lists common technical limitations of the mobile platform that derive from the wide scale of hardware specifications of the devices manufactured by various companies. The author reflects on the effects of these limitations on the audio design process, introduces audio middleware solutions that have been developed to ease the process of designing and implementing audio and covers basic programming terminology and programming methods used in audio implementation. Finally, the author presents an adaptive music system which he developed specifically for the game.

Keywords	Sound design, game audio, middleware, adaptive music

Sisällys

1 Johdanto

Videopelit ovat kulkeneet pitkän tien 70-luvulta tähän päivään. Ensimmäiset niistä olivat visuaalisesti ja äänellisesti hyvin yksinkertaisia. Klassisessa Pong-pelissä peliruutua kansoitti yhteensä kolme valkoista viivaa ja yksi pelipalloa esittävä objekti. Vähitellen teknologisten edistysaskelten myötä pelit ovat muuttuneet immersiivisemmiksi ja monipuolisemmiksi. Kun ensimmäisistä konsolipeleistä oli mahdollista laskea ruudulla käytetyt pikselit paljaalla silmällä, uusimpien pelien kuvakaappauksia voisi melkein erehtyä luulemaan valokuviksi. On hämmästyttävää, kuinka lyhyessä ajassa videopelit ovat kehittyneet pienen yleisön kuriositeettiviihteestä suurteollisuudeksi, joka tuottaa vuosittain kymmeniä miljardeja euroja.

Myös pelaamisen kulttuuri on muuttunut hyvin olennaisesti vuosikymmenien varrella. Aikanaan 80-luvulla videopelejä pelattiin hyvin pitkälti pelihalleissa, ja pelikonsoleita ostetiin lapsille viihdykkeeksi olohuoneisiin. Tietokoneilla pelaamista harrastivat vain asiaan tarkoin vihkiytyneet. Nykypäivänä sen sijaan pelit tavoittavat enemmän ihmisiä kuin koskaan aiemmin, ikään tai sukupuoleen katsomatta. Kehityskulkuun on varmasti vaikuttanut jo ennestään olemassa olevien pelialustojen rinnalle kehittynyt uusi alusta: mobiililaitteet.

1.1 Mobiilipeleistä yleisesti

Puhelimilla pelattavat pelit ovat käyneet läpi hyvin samanlaisen kehityskaaren kuin konsoli- ja pc-pelit. Ensimmäiset mobiilipelit olivat lähtökohtaisesti yhtä yksinkertaisia alkuperäisen Pong-pelin kanssa (esim. Nokian klassinen matopeli). Vuosien myötä puhelinten numeronäppäimistöt ovat vaihtuneet kosketusnäyttöihin, jotka pystyvät parhaimmillaan toistamaan HD-tasoista kuvaa.

Mahdollisuudet pelinkehitykselle ovat kasvaneet huomattavasti teknisten spesifikaatioiden muututtua tehokkaammiksi. Vähitellen sekä puhelinvalmistajat että pelinkehittäjät ovat ymmärtäneet mobiililaitteiden potentiaalin, jonka myötä mobiilipeleille on kehitetty omia markkinapaikkoja. Näistä tunnetuimmat lienevät Googlen Play-kauppa ja Applen App Store.

1.2 Kasuaalipelit

Viime vuosina on alettu puhumaan mobiilialustoille julkaistavista valtavirralle suunnatuista peleistä kasuaalipeleinä. Kasuaalipeli on yleinen määritelmä pelille, jota pelataan yleensä lyhyissä sykäyksissä. Kasuaalipelit on suunnattu suuren yleisön satunnaiseksi peli-iloksi, ja ne kattavat laajan kirjon erilaisia peligenrejä. Kasuaalipeli on hyvin usein ilmainen, mutta pelin sisällä on yleensä mahdollisuus maksaa esimerkiksi pelimaailmassa etenemisessä auttavista erikoisesineistä. (Horowitz & Looney 2014.)

Casual Connectin julkaiseman raportin mukaan vuonna 2013 kasuaalipelit tuottivat USA:n, Länsi-Euroopan ja Oseanian alueilla yhteensä 38 miljardia dollaria, ja samojen alueiden yhteenlasketun tuoton on projisoitu vuonna 2017 olevan 43,8 miljardia dollaria (Casual Games Association 2014). Kasuaalipelien suosio on siis jatkuvassa kasvussa, ja niitä tehdäänkin suosionsa takia jatkuvasti enemmän.

1.3 Opinnäytetyön tavoitteet

Olin mukana toteuttamassa opinnäytetyöni teososana Diamond Duo -nimistä mobiilipeliä SongHi Entertainment Oy -nimisessä pelialan yrityksessä. Kyseinen peliprojekti oli äänisuunnittelijana minulle ensimmäistä laatuaan, joten koko projekti oli oppimisnäkökulmasta erittäin antoisa.

Tavoitteenani on tässä opinnäytteessä tarkastella mobiilipelin äänisuunnittelutyötä teknisestä näkökulmasta. Käyn läpi mobiilipelin äänisuunnittelutyössä huomioitavia teknisiä rajoitteita. Aion valottaa teknistä prosessia käsittelemällä peliprojektin tuotantovaiheita sekä peliin valittua teknologiaa. Pyrkimykseni on selkeyttää työni vaiheita niin itselleni kuin sellaisille, joilla ei välttämättä ole kokemusta pelien parissa työskentelystä. Sivuan taustoittavasti peliäänen funktioita ja pelisuunnitteluun liittyvää termistöä.

Käyn myös läpi omaa oppimisprosessiani elokuvaäänen opiskelijana, jolla ei ollut peliprojektin alkamishetkellä käytännön kokemusta peliääniteknologian kanssa työskentelystä. Lähteinä opinnäytetyössäni käytän pääasiallisesti kahta peliääneen keskittyvää perusteosta, joissa käsitellään peliääntä teoreettisesta, teknisestä ja taiteellisesta näkökulmasta.

Seuraavassa luvussa syvennyn taustoittamaan projektia ja projektin lähtökohtia, omia tavoitteitani peliprojektissa ja muutamia yleisiä vallitsevia ajatuksia peliäänen funktioista ja kategorioista.

2 Projektin taustoitus

Aloittaessani Diamond Duon äänisuunnittelijana minulla oli vähänlaisesti kokemusta peliäänisuunnittelusta. Osallistuin syksyllä 2013 Metropolian 3D-animoinnin ja visualisoinnin opiskelijoiden peliprojektin “Dream Delusions” -äänisuunnittelutyöhön, johon toteutimme yhdessä kurssitovereideni kanssa äänitehosteet. Pääsin projektin yhteydessä tutustumaan pelissä käytettyyn Unity -pelimoottoriin, jota käytettiin myös tämän opinnäytetyön teososan toteutuksessa. Kerron Unity-moottorista enemmän luvussa “Äänen tekninen toteutus”.

Tietouteni ohjelmoinnista ennen projektin alkamista oli vähäistä. Olin aiemmin kehittänyt freelancerina toimiessani web-sivuja, joten minulla oli hieman tietämystä HTML- ja PHP-kielien käsittelystä. Edellä mainitut kielet ovat kuitenkin hyvin erilaisia kuin peliohjelmoinnissa käytetyt ohjelmointikielet, joten niiden osaamisesta ei liiemmin ollut hyötyä.

2.1 Yrityksestä

Pelin on toteuttanut SongHi Entertainment Oy, jonka pääasiallisena äänisuunnittelijana ja pelisäveltäjänä olen toiminut kesästä 2014 lähtien. Yrityksen pelituotantotiimissä on useita alan ammattilaisia – työryhmän jäsenien entisiä työpaikkoja ovat mm. ranskalainen pelituotantoyhtiö Ubisoft, Rovio ja Digital Chocolate.

SongHi Entertainment Oy perustettiin vuonna 2007. Ensimmäisenä projektinaan yritys teki selainpohjaista musiikintekopeliä, joka suunniteltiin helppokäyttöiseksi myös sellaisille ihmisille, joille musiikinteko ei ollut entuudestaan tuttua. Tuotetta on sittemmin jatkokehitetty opetustyökaluksi koulujen musiikkitunneille. Yritys on ennen Diamond Duoa kehittänyt yhden mobiilialustoille suunnatun kasuaalipelin, nimeltään “Melody Monsters”.

2.2 Diamond Duo -pelin perusidea ja mekaniikka

Pelissä pelaajan tarkoitus on Robin Hoodin omaisesti ryövätä kuvitteellisen kansan ylimystöltä jalokiviä ja aarteita. Pelaajahahmona toimivat Fay-niminen fretti ja hänen tohelo apurinsa. Heidän perässään kulkee kömpelö seriffi Barx, jonka ainoana tavoitteena on saada päähenkilöt kiinni. Pelissä liikutaan pelimaailmassa kentästä toiseen aarteita keräten, ja pelaajan tavoitteena on saavuttaa kenttäkohtaisesti määritelty vähimmäispistemäärä kentän läpipääsemiseksi.

Diamond Duo edustaa tyypiltään kasuaalipeliä. Siitä on pyritty muiden samankaltaisten pelien tavoin tekemään helposti lähestyttävä kokonaisuus sellaiselle ihmiselle, joka ei välttämättä muuten harrasta pelaamista. Siihen on lisätty sosiaalisia ominaisuuksia, joiden avulla pelaaja voi seurata etenemistään suhteessa muihin ystäviinsä. Pelaaja voi esimerkiksi yhdistää pelin Facebook-tiliinsä.

Diamond Duon pelimekaniikka on variaatio yleisesti kasuaalipeleissä esiintyvästä ”match-3”-mekaniikasta (”Tile-matching video game”) jota on käytetty muun muassa sellaisissa kasuaalipeleissä kuten Bejeweled sekä Candy Crush Saga (Wikipedia 2015).

Pelissä kerätään pisteitä yhdistämällä kolme tai useampi samanvärinen timantti keskenään. Yhdistäminen tapahtuu napauttamalla yksittäistä jalokiveä, jolloin valittu jalokivi poistuu pelialueelta. Poistuneen jalokiven yllä olleet jalokivet lastkeutuvat ylhäältä alas, saaden pelilaudalla aikaan mahdollisen pistetilanteen (ks. liite 1). Jokainen poistettu jalokivi vähentää pelaajalta siirron. Siirtojen määrä on määritelty kenttäkohtaisesti. Pisteitä kerätään niin kauan kunnes kentälle määritelty vähimmäispistemäärä on saatu täyteen, jolloin pelaaja läpäisee kentän ja pääsee siirtymään seuraavaan kenttään. Jos pelaajalta loppuu siirrot ennen vähimmäispistemäärän saavuttamista, pelaaja häviää ja joutuu aloittamaan kentän alusta.

Perusmekaniikan lisäksi pelissä on mahdollista kääntää koko pelikenttää yhdeksänkymmentä astetta vasemmalle tai oikealle, mikä avaa pelaajalle uusia tulokulmia tilanteissa, joissa vaikuttaa mahdottomalta tehdä suurilla pistemäärillä palkitsevia siirtoja. Peliruudulla on tavallisten jalokivien lisäksi erikoisjalokiviä, joilla voi tyypistä riippuen puhdistaa yhdellä siirrolla suuriakin alueita peliruudulta. Peliä vaikeu-

tetaan paikoitellen kivillä, joita ei voi poistaa peliruudulta muilla keinoilla kuin erikoisjalokivillä tai erikoiskyvyillä.

Erikoiskykyjä pelissä ovat jousi, jalokivipussi ja pommi. Jousella on mahdollista poistaa ruudulta yksi jalokivi ilman että se kuluttaa pelaajalta siirtoa. Jalokivipussi kerää kaikki pelaajan valitseman värin väriset jalokivet pois ruudulta. Pommi taas räjäyttää pelaajan määrittämästä kohdasta laskien ristin muotoisen alueen ja kerää kaikki alueelle osuvat jalokivet pelaajan haltuun.

Pelin ensimmäisessä julkaisussa kenttiä oli yhteensä 40 – projektin suunnitelmassa oli ennalta määrätty, että kenttiä tullaan lisäämään säännöllisin väliajoin päivitysten yhteydessä, jotta pelin elinikää saataisiin pidennettyä aina tarpeen mukaan.

2.3 Pelin tuotantovaiheet

Peliäänituotannon prosessi muistuttaa joiltain osilta elokuvaäänen tuotantoprosessia. Molemmissa käytetään hyvin pitkälti samoja äänitystekniikoita, tuotantovälineitä sekä äänen jälkikäsittelyyn tarkoitettuja ohjelmistoja. Toisaalta myös erot näiden kahden eri tuotantolajin välillä ovat huomattavia. Äänituotanto on elokuvaproduktiossa hyvin jälkityöpainoitteinen prosessi, jossa työskennellään ajallisesti ja sisällöllisesti lukitun kuvan kanssa. Peliproduktion aikana sen sijaan ollaan usein tekemisissä keskeneräisten visuaalisten elementtien kanssa, jotka saattavat kehittyä dramaattisestikin tuotannon etenemisen myötä. (Collins 2007.)

Olennaista ennen äänituotannon aloittamista on määrittää pelin genre ja teema, jotta äänisuunnittelu voidaan ottaa osaksi pelintekoprosessia jo lähtömetreiltä asti (Collins, 2007). Diamond Duon kohdalla tiesin kyseessä olevan puzzle-peli, koska yksi ohjelmoijista oli työstänyt pelimekaniikkaa jo prototyyppivaiheessa.

Diamond Duon ennakkotuotannon aikana sain aluksi tehtäväkseni tehdä tuottajille selvitys tarvittavasta ääniefektien ja musiikin määrästä. Käytin pelistä tehtyä prototyyppiä apunani ääniefektien määrän tarpeen arvioinnissa. Lisäksi tein dokumentaation pelin äänellisistä tavoitteista ja budjetillisista tarpeista. Yksi tavoitteista oli säveltää peliin musikkia, joka valmistuttuaan äänitettäisiin studiossa orkesteri-instrumenteilla soitettuina. Toisena tavoitteenani oli tehdä musiikista olennainen osa pelikokemusta, mikä auttaisi pelaajaa saamaan yhteyden pelin esittelemään tarumaailmaan.

Tuotannon käynnistyttyä Diamond Duossa käytettiin alkuun placeholder-taidetta, jonka avulla visualisoitiin pelin kannalta olennaisia elementtejä, kuten käyttöliittymää ja kerättäviä peliobjekteja. Pelin lopullinen teema valittiin hyvin paljon myöhemmin tuotannon alkamisen jälkeen, joten teeman valitsemiseen asti jouduin muun työryhmän tavoin työskentelemään yksinomaan placeholdereiden armoilla.

Varsinaiseen peliin lopulta päätyvän äänituotannon aloitin joulukuun 2014 aikana. Pyrkimyksenä oli aluksi täyttää peli olennaisimmilla ääniefekteillä. Näiden alussa tehtyjen efektien pohjalta siirryin sittemmin iteroimaan ja tekemään tehosteista erilaisia versioita, kunnes saavuttaisin itseäni miellyttävän lopputuloksen. Iterointi- ja kokeiluprosessia helpotti huomattavasti käytetyt ohjelmistot, joista kerron lisää luvussa "Äänen tekninen toteutus."

Halusin erityisesti kiinnittää tuotannossa huomiota peliäänen funktioihin. Koin, että tarkalla ja harkitulla äänityöllä olisi mahdollista elävöittää ja parantaa pelikokemusta olennaisesti. Tavoitteenani oli osoittaa, että huolellisella äänisuunnittelulla on merkitystä jopa Diamond Duon kaltaisessa kasuaalipelissä.

2.4 Peliäänen funktiot ja kategoriat

Äänen rooli vaihtelee suunnattomasti riippuen pelin tyypistä ja pelialustasta. Ääriesimerkkejä käyttäen odotukset ongelmanratkontapelin äänimaailmalle ovat hyvin erilaiset kuin kolmiuloitteiselle ammuskelupelille.

Koska konsoli- ja pc-pelejä pelataan pääasiassa kotioloissa joko TV:n tai tietokonenäytön välityksellä ilman ulkoisia häiriötekijöitä, niihin liittyvät pelikokemukset poikkeavat huomattavasti monien puhelimella pelattavien pelien tarjoamista kokemuksista. Mobiilipelejä pelataan hyvin usein julkisilla paikoilla, kuten joukkoliikennevälineissä tai kahviloissa. Jos käyttäjällä ei ole edellä mainituissa paikoissa mukanaan kuulokkeita, äänet jätetään kanssaihmiset huomioimiseksi pois päältä.

Esimerkiksi Epic Gamesin tuottamien Unreal-pelien parissa työskennellyt säveltäjä ja äänisuunnittelija Alexander Brandon toteaa Essential Guide to Game Audio –teoksessa seuraavaa: "Oli hauskuus FPS-pelin keskiössä tai ei, loppujen lopuksi immersio on tärkeintä" (Horowitz & Looney 2014). Voidaan todeta, että pc- ja konsolipeleissä ääni

on immersiivisyyden osalta tärkeässä asemassa, kun taas tavanomaisissa mobiilipeleissä äänellä on kenties olennaisempi funktio informaation välittämisessä. Poikkeuksia tietenkin löytyy alustalla kuin alustalla. Älypuhelimet ovat kehittyneet huomattavasti vuosien aikana ja joidenkin pelien yksinkertaisuus ei tätä nykyä ole niinkään alustan rajoitteiden sanelemaa kuin tietoinen valinta suunnittelussa.

Yksinkertaisin tapa kategorisoida ääniä juontaa juurensa elokuvamaailmasta, diegeettisiin ja ei-diegeettisiin ääniin. Pelikontekstissa diegeettisen äänen piiriin katsotaan kuuluvan esimerkiksi pelimaailmassa sijaitsevian objektien ja hahmojen äänet. Ei-diegeettisiin ääniin voidaan sisällyttää mm. voiceover-äänet ja pelin taustamusiikki (Horowitz & Looney, 2014, s. 76). Miten ääni tulisikaan kategorisoida esimerkiksi silloin, kun kyseessä on immersiivisyyteen tähtäävän ammuskelupelin sijasta koskетusnäytöllä pelattava älypeli?

Videopeliäänen funktioista on käyty viime vuosina paljon keskusteluja, ja syvempi analyysi funktioista vaatisi paljon mittavampaa perehtymistä kuin mihin minulla on ollut mahdollisuus tämän opinnäytetyöprojektin puitteissa. Esittelen seuraavaksi muutaman yleiskattavan huomion peliäänen ja -musiikin rooleista.

Dave Russell (2012) määrittelee peliäänen tärkeimmät funktiot seuraavasti:

1. Palaute - ääni antaa palautetta pelaajalle hänen tekemistään toiminnoista.
2. Informaatio - ääni antaa pelaajalle tietoa pelimaailmasta ja peliympäristöstä.
3. Saavutukset - Ääni on osana ”saavutusjärjestelmää”, jolla välitetään pelaajalle informaatiota hänen onnistumisistaan. Tästä esimerkkinä alkuperäinen Tetris, jossa rivi poistuu aina välkkyvän valon ja ääniefektin saattelemana.
4. Realismi ja immersio - Ääni tuo pelimaailman lähemmäksi pelaajaa, ja auttaa pelaajaa samastumaan pelin tapahtumiin.
5. Tunnelma ja rytmi - Ääni määrittää usein pelin tunnelmaa ja rytmiä musiikin muodossa.

Horowitz ja Looney (2014) sen sijaan esittävät, että pelimusiikilla on tarkoituksena:

1. Määrittää pelin kokonaistunnelma
2. Informoida pelaajaa siitä, mihin aikaan ja sijaintiin peli sijoittuu
3. Informoida pelaajaa siitä, missä lokaatioissa ja paikoissa pelissä liikutaan.

4. Auttaa pelaajaa identifioimaan pelimaailman hahmoja
5. Määrittää koko pelikokemuksen rytmi
6. Vahventaa immersion kokemusta

Päämääränäni oli hyödyntää edellä mainittuja huomioita äänen ja musiikin rooleista peleissä. Ensikertalaisuuteni peliäänisuunnittelijana tuntui hyvältä tilaisuudelta kokeilla käytännössä näitä esitettyjä ajatuksia, ja katsoa miten niiden huomioiminen vaikuttaisi suunnitteluprosessiin.

Seuraavassa luvussa otan esille huomioita mobiilialustojen teknisistä rajoitteista, ja annan esimerkkejä siitä miten nämä rajoitteet vaikuttavat äänisuunnittelutyöhön.

3 Mobiilipeleissä huomioitavat tekniset rajoitteet

Elokuvaa äänisuunnitellessa ei koskaan kohtaa tilannetta, jossa tulisi ennen suunnittelutyötä ottaa huomioon vaikkapa käytettyjen äänitehosteiden tai ambienssiraitojen määrä muusta kuin esteettisistä syistä. Finaalimiksaus vie äänitiedostona sen verran tilaa kun on tarve, lopullisesta raakaformaatissa olevasta videoleikkauksesta puhumattakaan. Elokuvaäänisuunnittelijan ei myöskään tarvitse huolehtia siitä, riittävätkö eri teattereiden DCP- tai blu-ray-toistimien tehot tuhansia räjähdyksiä ääniraidallaan hyödyntävän elokuvan näyttämiseen.

Tilanne on toisenlainen pelinkehitystyössä, etenkin mobiilialustoilla: lähtökohtaisesti pelin kokoa ja laiteresurssien, kuten prosessorin ja muistin käyttöä tulisi tarkkailla koko tuotteen elinkaaren ajan.

Halvimpien ja kalliimpien mobiililaitteiden laitespesifikaatioilla voi olla suuriakin eroja. Erojen takia on tärkeää pyrkiä optimoimaan peli niin, että se toimii vanhemmillakin laitteilla. Pelinkehityksessä pyrkimys on loppujen lopuksi varmistaa, että mahdollisimman suurella joukolla ihmisiä on valmiudet pelata peliä, laitteella kuin laitteella.

Suurin osa optimointityöstä on yleensä ohjelmoijien vastuulla. Äänisuunnittelijan ominaisuudessa olen kuitenkin vastuussa pelin äänellisten elementtien optimoinnista. Syvennyn optimointiprosessiin tarkemmin kappaleessa “Väliohjelmiston hyödyt.”

3.1 Tilan käyttö

Mobiilialustoille suunnatussa pelinkehityksessä on erittäin tärkeää huomioida tilan käyttö. Mobiililaitteista löytyy huomattavasti vähemmän prosessoritehoa, muistia ja kiintolevytilaa kuin esimerkiksi pöytäkoneista. Tästä syystä pelin sisältämien tiedostojen koko on pidettävä niin pienenä kuin mahdollista, jotta koko pelin koko pysyy hallittuna. (Horowitz & Looney, 2014). Esimerkiksi Applen asettama yläraja Appstoresta 3G-yhteydellä ladattavien pelien ja applikaatioiden koolle oli vielä ennen iOS7-käyttöjärjestelmän julkaisua 50 megatavua – tämän opinnäytetyön kirjoitushetkellä yläraja on 100 megatavua. (AppleInsider 2013.)

Tilan käytölle asetetut rajoitteet asettavat pelintekijöille useita haasteita. Mobiililaitteiden spesifikaatioissa on paljon variaatiota: esimerkiksi iPhone 4S:n resoluutio on 960 x 640, kun taas iPhone 6:n resoluutio on 1334 x 750 (Apple 2014). Pelistä voi olla saatavilla vain yksi versio kerrallaan käyttöjärjestelmää kohden, jolloin resoluutioiden varianssiin täytyy vastata sisällyttämällä peliin sen graafisista elementeistä useita eri laatuisia versioita. Tämä nostaa vääjämättä lopullisen pelin kokoa suuremmaksi, kuin sen tarvitsisi olla esimerkiksi hieman vanhemmilla laitteilla. Peli ohjelmoidaan lataamaan laitekohtaisten spesifikaatioiden mukaan määritellyt versiot graafisista elementeistä. Myös pelimoottorille joudutaan varaamaan osansa pelin kokonaiskoosta.

Edellä mainitut seikat vaikuttavat luonnollisesti myös äänelle jäävän tilan suuruuteen. Päämääränä on tehdä hyvänkuuloinen peli, mutta minimaalisella tilankäytöllä.

3.2 Resurssien käyttö

Mikä tahansa mobiililaitteella avattu prosessi käyttää laitteen muistia ja prosessoria. Ohjelmiston raskaus vaikuttaa siihen, kuinka suuren prosenttiosuuden se käyttää vapaana olevasta muistista ja prosessointitehosta. Esimerkiksi 3D-tekniikkaan pohjautuva ammuskelupeli käyttää huomattavasti enemmän muistia ja prosessorin tehoa kuin vaikkapa yksinkertainen 2D-elementeistä koostuva matopeli. Muistista ja prosessorista käytössä oleva prosenttiosuus taas vaikuttaa suoraan akunkestoon. Päämääränä on pitää sekä muistin että prosessorin käyttö kurissa niin, että ohjelmisto toimii sulavasti eikä kuluta kohtuuttomasti käyttäjän laitteen akkua. (Unity Technologies 2015.)

3.3 Äänentoisto

Äänisuunnittelussa on otettava huomioon, minkälaisella laitteella ääniä kuunnellaan. Hyvin usein mobiililaitteiden äänentoisto poikkeaa huomattavasti tavallisesta kaiutinkuuntelusta. Pieneen älypuhelimeen on fyysisesti mahdoton pakata sellaisia kaiutinelementtejä, jotka pystyisivät toistamaan tavallisten pöytäkaiuttimien tavalla ääniä laajalla taajuuskaistalla. Matalimmat toistotaajudet vaihtelevat laitekohtaisesti, mutta suurimmassa osassa laitteita matalin kuultava ääni sijoittuu n. 300 – 500 Hz välille. Pahimmassa tapauksessa äänet kuulostavat studiokuuntelussa erittäin hyvältä, mutta kännykästä toistettaessa puuroiselta ja epäselvältä.

Lisäksi tilannetta mutkistaa käyttäjän mahdollinen kuulokkeiden käyttö. Jos äänet suunnittelee puhelimen kaiutintoistoa varten niin, että niihin ei pakkaudu liikaa ongelmallisia taajuuksia, saattavat äänitehosteet kuulostaa kuulokkeilla toistettaessa ohueilta.

Näihin ongelmiin vastataan yleensä tekemällä peliäänelle eri äänentoistolaitteita varten yksilölliset ”miksaukset.” Kun peli tunnistaa äänentoistolaitteen, ääniraita toistetaan ennalta asetetun taajuuskorjainefektin kautta. Taajuuskorjaimella on mahdollisuus poistaa tukkoisia taajuuksia äänentoiston tapahtuessa puhelimen kautta. Korjaus poistetaan käytöstä, kun äänentoisto siirtyy puhelimen sisäänrakennetusta kaiuttimesta kuulokkeisiin.

4 Peliäänen tekninen toteutus

Käyn seuraavaksi läpi äänen toteutuksellisen prosessin ääniefektin luomisesta peliin implementointiin asti. Vaikka esitän prosessin tässä opinnäytetyössä lineaarisena tapahtumajaksona, peliäänen tekninen toteutus voi usein olla prosessina hyvinkin dynaaminen. Joskus äänet on saatu oikeille paikoilleen pelin sisällä, mutta muutokset koodissa ja peliympäristössä saavatkin aikaan tilanteen, jolloin äänet eivät enää täsmääkään ruudun tapahtumien kanssa. Tärkeää on siis jatkuvasti pitää huolta, että pelin pysyy kokonaisuutena eheänä myös äänellisesti. Pelinkehitys on pitkä prosessi, jonka aikana ennalta tehtyjen suunnitelmien mukaiset toteutukset saattavat muuttua esimerkiksi pelaajatestauksen myötä.

4.1 Pelimoottori

Diamond Duo on toteutettu Unity-pelimoottorilla. Unity on yleisimmille pelialustoille suunnattu pelinkehitysohjelmisto, jota SongHi Entertainment Oy on käyttänyt aiemman Melody Monsters-pelinsä kehitystyössä. Pelitiimi käytti Melody Monsters-pelissä myös Unitylle ohjelmoituja lisätyökaluja, esimerkkinä Tasharen Entertainmentin kehittämä käyttöjärjestelmätyökalua NGUI (Next-Gen UI.) Näiden valmiiksi hyväksi havaittujen lisätyökalujen käyttöönotto Diamond Duossa nopeutti pelinkehityksen aloitusta.

4.2 Väliohjelmistot

Wikipedian mukaan väliohjelmisto on “ohjelmisto, joka suorittaa jotain tiettyä yksittäistä tehtävää jonkin suuremman ohjelman sisällä” (Wikipedia 2015). Peliäänen tapauksessa se tarkoittaa ohjelmistoa, jolla hallitaan kaikkia ääneen liittyviä toimintoja, kuten tiedostojen hallintaa, efektointia, toistoa ja kompressointia. Äänen väliohjelmistoja alettiin kehittämään, koska ääntä haluttiin kontrolloida enemmän ja tarkemmin kuin aiemmin oli pystytty. Väliohjelmistojen tarpeellisuutta on perusteltu pelien interaktiivisuudella: koska pelit ovat interaktiivinen media, myös äänellisten elementtien tulisi olla interaktiivisia. (Horowitz & Looney 2014.)

Ensimmäinen syy väliohjelmiston käyttöönottoon Diamond Duon kehitystyössä oli kokemattomuuteni peliäänen parissa työskentelyn saralla. Prioriteettinani oli peliprojektin yhteydessä kehittää omia taitojani niin äänisuunnittelun kuin äänen implementoinnin osalta. Unityssä on sisäänrakennettu mutta rajattu toiminnallisuus äänen ja musiikin hallinnalle. Jos yhdelle toiminnalle halutaan esimerkiksi kolme äänellistä variaatiota, tulisi äänitiedostot ensin implementoida Unity-projektiin, ja tämän jälkeen ohjelmoida pelin koodiin toiminto, joka hakisi satunnaisesti yhden näistä kolmesta tiedostosta toistoa varten. Väliohjelmistolla olisi mahdollisuus toteuttaa tälläinen äänen satunnaistoisto ilman erillistä ohjelmointia.

Pienessä työryhmässä on resurssien hallinnan kannalta tehokkaampaa, että äänisuunnittelusta vastaavalla ihmisellä on vastuu äänitehosteista ja musiikista niiden peliin viemiseen asti. Kun äänisuunnittelija vastaa kokonaisuudessaan äänestä sen tuotan-

nosta implementointiin, pelin ohjelmoinnista vastaaville ihmiselle jää enemmän aikaa keskittyä pelin ominaisuuksien ja mekaniikan kehittämiseen.

Tässä kappaleessa mainituista syistä ehdotin pelituottajille väliohjelmiston käyttöönottoa Diamond Duon kehitystyössä. Äänen hallintaan tarkoitettuja väliohjelmistoja on olemassa muutama, joista käytän tyyppiesimerkkeinä projektissa käyttämääni Firelight Technologiesin kehittämää FMOD Studiota, sekä Audiokineticin kehittämää väliohjelmistoa WWiseä (Wave Works Interactive Sound Engine). Molempia mainitsemistani esimerkeistä käytetään suuressa mittakaavassa niin PC- ja konsolirintamalla kuin mobiilipeleissäkin.

4.2.1 FMOD Studio

Päädyin käyttämään Diamond Duossa FMOD Studiota lyhyen kokeilujakson perusteella. Kyseisen väliohjelmiston kehittäjä Firelight Technologies julkaisi ensimmäisen versionsa FMODista n. 15 vuotta sitten. FMOD Studio on uusin versio tästä väliohjelmistosarjasta.

FMOD Studion käyttöliittymä on hyvin lähellä esimerkiksi Avid Pro Toolsin käyttöliittymää (ks. liite 2). Äänitiedostojen käsittely tapahtuu samalla tavalla kuin Pro Toolsissa, siirtämällä äänitiedosto halutulle raidalle. Kun valittu äänitiedosto on asetettu raidalle, sen sävelkorkeutta tai äänenvoimakkuutta voidaan muuttaa. Ääntä voi myös efektoida reaaliaikaisesti ohjelmiston omilla virtuaaliefekteillä. Lisäksi FMOD tarjoaa mahdollisuuden käyttää tempomerkintöjä ja tahtilajeja musiikin saumatonta uudelleentoistoa varten. Käyttöliittymän ymmärrettävyys ja tuttuus helpotti siirtymää elokuvan lineaarisesta editointitavasta peliprojektin dynaamisempaan lähestymistapaan.

Firelight Technologies on kehittänyt FMOD Studiolle oman integraatiopaketin, jonka avulla ohjelmisto oli helppo ottaa käyttöön Unityn sisällä.

4.2.2 Väliohjelmiston käyttöönoton hyödyt

Yksi suuri syy käyttää väliohjelmistoa pelimoottorin oman äänifunktionaalisuuden sijasta on väliohjelmiston mahdollistamat äänen hallintakeinot. Diamond Duon äänisuunnitelman mukaisesti halusin pelin äänien välittävän selkeää informaatiota pelaajalle

hänen onnistumisistaan. Päätin esimerkiksi käyttää pelissä ääniefektin sävelkorkeuden nostamista hyödyksi tilanteissa, jossa pelaaja laukaisee suuria pistemääriä kerryttävän ketjureaktion. Tavanomaisesti äänitiedoston sävelkorkeuden nostaminen tai laskeminen olisi ohjelmoitava erikseen, mutta yleisesti väliohjelmistoista löytyy tämänkaltainen funktionaalisuus sisäänrakennettuna ominaisuutena.

Äänen väliohjelmiston käyttö voi myös usein olla helpottava tekijä, kun peliä sopeutetaan sellaiselle alustalle, jolle sitä ei ole alunperin kehitetty. Diamond Duo on kehitetty iOS-käyttöjärjestelmälle, mutta esituotannossa tehtiin päätös tuottaa pelistä myöhemmässä vaiheessa myös Android-versio.

Esituotantovaiheessa toin tuottajien tietoon, että FMOD Studion käyttöönotto peliprojektissa helpottaisi tulevan Android-version kehitystyötä. Sekä FMOD että WWise sisältävät valmiiksi alustakohtaiset asetukset, joiden avulla äänen funktionaalisuus voidaan siirtää sellaisenaan alustalta toiselle. Tulevan Android-version kehitystyön yhteydessä ohjelmoijien on siis mahdollista tehdä Diamond Duosta äänen kannalta täysin identtinen iOS-version kanssa.

Tilan säästämiseksi väliohjelmistot tarjoavat usein mahdollisuuden kompressoida pelissä käytetyt äänitiedostot pienempään kokoon. Diamond Duossa suurin osa pelin äänitehosteista on viety FMOD Studioon WAVE-formaatissa yksikanavaisena, näytteenottotaajuutena 44.1kHz ja bittisyvyytenä 16 bittiä.

Yksi olennainen osa äänikirjaston rakennusprosessia on mahdollisuus organisoida käytetyt äänet eri kategorioihin. Kun kaikki äänitapahtumat ovat siistissä järjestyksessä, on helppo tarpeen tullessa muokata olemassaolevia ääniä tai vaihtaa niitä uusiin. FMOD Studion käyttöjärjestelmä antaa luoda kansioita ja alikansioita äänitapahtumille, jotka voidaan nimetä halutulla tavalla.

Viedessäni ulos FMOD Studiossa suunnittelemani äänikirjaston pystyn määrittämään haluamani kompressiointiasetukset sekä äänille että musiikille. Äänisuunnittelija voi halutessaan määrittää eri ääniefekteille eri kompressiointiasetukset. Esimerkiksi Diamond Duon karttanäkymässä (johon pelaaja palaa aina läpäistyään kentän) soi taustalla pitkä, yhtäjaksoinen atmosfääriraita. Pakkaamattomana WAVE-tiedostona mainittu atmosfääriraita veisi 3 megatavua, mutta Ogg Vorbis -pakkauksen jälkeen tiedoston koko pienenee 10-kertaisesti 300 kilotavuun. Toisaalta taas pelissä selkeästi esillä

olevissa äänitehosteissa päätin käyttää pienempiä kompressointiasetuksia. Koin, että jatkuvasti toistuvat äänet huonolaatuisessa muodossa saattaisivat tuntua pelaajasta häiritsevältä.

Äänen siirtäminen väliohjelmiston hallintaan antaa mahdollisuuden seurata tarkasti, mitkä elementit peliäänestä vievät eniten prosessoritehoja ja muistia. Sekä FMOD Studiossa että WWisessä tulee mukana mahdollisuus profiloida äänen osuutta käytetyistä laiteresursseista.

4.3 Äänen ohjelmointi Unityssä

Projektin aikana opettelin työryhmän ohjelmoijien avulla äänen implementointiprosessissa vaadittuja ohjelmointitaitoja. Unityssä ohjelmointikielenä käytetään joko JavaScriptiä (JS) tai C#-kieltä. Diamond Duoon valittiin ohjelmointikieleksi C#, koska työryhmän ohjelmoijat olivat työskennelleet kyseistä kieltä käyttäen myös aiemmin. Koska olen melko kokematon ohjelmoinnin saralla, käyn tässä kappaleessa läpi hyvin yksinkertaistetusti Diamond Duon ääntä koskevaa ohjelmointiprosessia.

C# on Microsoftin kehittämä oliopohjainen ohjelmointikieli, joka muistuttaa syntaksiltaan C++- ja Java-kieliä. Oliolla tarkoitetaan ohjelmiston perusyksikköä, joka sisältää joukon loogisesti yhteenkuuluvaa tietoa ja toiminnallisuutta (Wikipedia 2013). Jotta olioita olisi helpompi järjestellä ja kutsua käyttöön, ne järjestellään luokkiin. Mika Kolari kirjoittaa luokista ja olioista näin:

> Luokat ovat olio-ohjelmoinnin perusosia. Ne kokoavat yhteen jonkin tietyn toiminnallisuuden, jota voidaan käyttää uudestaan. Oliot ovat luokkien ilmentymiä (instansseja) eli muuttujia, joiden tyyppi on jokin luokka. Jos kirja olisi luokka, voisi ”Sinuhe egyptiläinen” olla yksi olio. (Kolari 2011.)

Ääniefektien ja musiikin kutsumiseksi pelin sisällä minun tuli ensin luoda uusi luokka, johon tulisin keräämään kaikki äänen funktionaalisuuteen liittyvät oliot. Selkeyden vuoksi tämän luokan nimeksi annettiin Audiomanager – kuvaavan nimeämisen kautta muut ohjelmoijat tietäisivät luokan käyttötarkoituksen jo pelkästään nimen perusteella. Suurin osa Audiomanager-luokan olioista olivat tyypeiltään metodeja. Metodilla tarkoitetaan kokonaisuutta, johon kerätty joukko suoritettavia lauseita. Metodin sisälle

kerätyt lauseet voidaan kutsua yhdellä komennolla, sen sijaan että jokaista vaadittua lausetta kutsuttaisiin erikseen yksi kerrallaan.

FMOD Studiossa rakentamassani äänikirjastossa kaikki ääniefektit ja musiikit on jaoteltu tapahtumiin. Jokaista tapahtumaa voisi kuvata yhdeksi äänelliseksi kokonaisuudeksi; tapahtuman sisällä voi olla useampia raitoja, ja raidoilla voidaan käyttää reaaliaikaisesti prosessoriteholla laskettavia ääniefektejä (kuten kaikua tai taajuuskorjainta). Tapahtumien tarjoamia mahdollisuuksia hyödyntämällä FMOD Studiossa on mahdollista rakentaa hyvin monimutkaisiakin parametreilla ohjattuja kokonaisuuksia, joista kerron lisää kappaleessa “Parametrit”.

Luokan luomisen jälkeen minun tuli ohjelmoida referenssit FMOD Studiolla rakennetun kirjaston äätapahtumiin. Diamond Duon tapauksessa nämä referenssit sisälsivät tiedostopolun, jonka kautta ohjelmisto pystyy hakemaan tapahtumia äänikirjastosta. Referenssit toimivat siis kommunikaatioväylänä peliohjelmiston ja väliohjelmiston välillä.

Tyypillisin esimerkki Diamond Duossa käyttämästäni metodista on yksittäisen ääniefektin kutsuminen FMOD Studiossa rakentamastani äänikirjastosta. Ensin luon metodin ja nimeän sen käyttötarkoituksen mukaan (esim. “PlayLoseJingle”, soita häviömusiikki). Metodin sisällä käytän FMOD Studion Unity -integraatiossa määriteltyä komentolausetta “PlayOneShot”, jolla voidaan kutsua mitä tahansa pelin äänikirjastoon sisällytettyä tapahtumaa. Myös PlayOneShot-komento on itsessään referenssi metodiin, joka löytyy FMOD-integraatioon kuuluvasta luokkakirjastosta. PlayOneShot-komentolausekkeen sisällä käytän luomaani referenssiä äänikirjastossa sijaitsevaan “LoseJingle”-tapahtumaan.

Häviömusiikkia toistavan metodin rakentaminen Audiomanager-luokkaan ei yksistään riitä äänen soittamiseksi pelissä. Tavallisin tapa halutun äänen soittamiseen on etsiä jostain toisesta luokasta metodi, joka liittyy olennaisesti toistettavaan ääneen. Häviömusiikin tapauksessa minun tulee etsiä metodi, jota hyödyntäen pelin sisällä näytetään tekstibanneri “You Lost” (“Sinä hävisit”). Lisäämällä häviöbanneria kutsuvaan metodiin viittauksen häviömusiikkiin, musiikki toistuu samalla kun banneri tulee esille. “PlayOneShot”-komentolausekkeeseen sisältyy käsky vapauttaa äänentoistoon käytetyt muisti- ja prosessoriresurssit toiston päätyttyä, millä varmistetaan ettei äänitehosteet jää turhaan kuluttamaan laitteen tehoja.

4.4 Parametrit

FMOD Studiossa on mahdollista määrätä äänitapahtumalle erilaisia parametreja. Yksinkertaisin parametri on äänenvoimakkuus. Molemmille, sekä minimi- että maksimivoimakkuuksille, voidaan ensin määrittää tapahtumassa mikä tahansa viitteellinen desibeliarvo. Minimi- ja maksimivoimakkuuksien määrittämisen jälkeen annetaan minimivoimakkuudelle vastaava parametrin arvo 0, ja maksimivoimakkuudelle vastaava parametrin arvo 1. Parametrien käytöllä on tarkoitus nopeuttaa ja helpottaa äänenvoimakkuuden kontrolloimista dynaamisesti.

Kun peli lähettää tapahtumalle arvon 0, se toistuu määritellyllä minimivoimakkuudella, ja toisaalta saadessaan arvon 1, se toistuu määritellyllä maksimivoimakkuudella. Parametrille voidaan lähettää arvoja portaattomasti, eli se voi saada minkä tahansa arvon väliltä 0 ja 1. Tapahtuman äänenvoimakkuus määräytyy saman skaalan mukaisesti.

Luvussa “Väliohjelmiston hyödyt” mainitsin eräänä pelin tehokeinona keskeisten äänitehosteiden sävelkorkeuden muuttamisen pelitilanteissa, joissa pelaaja laukaisee suuria pistemääriä kerryttäviä ketjureaktioita. Sävelkorkeuden muuttaminen tapahtuu helpoiten määrittämällä halutuille parametrin arvoille tietyt sävelkorkeudet. Diamond Duon tapauksessa määritin ääniefektin sävelkorkeuden lähtöarvoksi lukuarvon 1. Jokainen tasaluku vastaisi yhden sävelaskeleen nostoa, jolloin parametrin arvo 2 nostaisi ääniefektin sävelkorkeutta yhdellä sävelaskeleella. Parametrin arvot haetaan sellaisesta luokasta, joka välittää informaatiota siitä kuinka mones pelaajasta riippumaton ketjureaktio peliruudulla on meneillään. Kun peliruudulla on meneillään vaikkapa ketjureaktion kolmas vaihe, saa ääniefekti arvon 3, jolloin sävelkorkeus on kaksi sävelaskelta lähtösäveltä korkeampi.

4.5 Musiikin toteutus

4.5.1 Sävellys ja äänitys

Ennen Diamond Duo-projektia olin säveltänyt pelin prototyyppiin luuppaavan taustamusiikin. Työryhmä tykästyi musiikissa käytettyyn melodiakulkuun niin paljon, että päätimme hyödyntää sitä myös projektin varsinaisessa tuotantovaiheessa. Valmiin sävellyspohjan käyttäminen nopeutti musiikin tuotantoprosessia huomattavasti.

Ennakkotuotantovaiheessa tutustuin monen kasuaalipelin ääni- ja musiikkitoteutukseen. Huomasin, että monessa pelissä oli päädytty ratkaisuun, jossa toistetaan yhtä n. 15 sekunnin mittaista musiikkikiertoa ilman variaatioita. Itseni haastamiseksi ja kehittämiseksi ehdotin Diamond Duon tuottajille, että pelin musiikkiraita toteutettaisiin vähemmän konventionaalisilla keinoilla.

Lähtökohdakseni otin ajatuksen adaptiivisesta musiikista. Adaptiivinen musiikki tarkoittaa musiikkia, joka reagoi pelaajan valintoihin ja toimiin (Horowitz & Looney, 2014.) Otin suunnitelmakseni rakentaa musiikkijärjestelmän, jonka avulla musiikin orkestraatio kasvaa kun pelaajalla menee pelissä hyvin, ja jolla saataisiin vähennettyä instrumentteja, kun pelaaja ei saa kerättyä tarpeeksi pisteitä. Halusin adaptiivisuudella vahventaa musiikin merkitystä ja tehdä siitä oleellisen osan pelikokemusta.

Toteutin ensimmäisen demon peliä varten tehdystä sävellyksestä yksistään Logic Pro X- digitaalityöasemalla. Käytin sävellystyössä ohjelmiston mukana tulleita orkesterisamplekirjastoja, jotka eivät sellaisenaan kuulosta järin aidoilta, mutta joita on nopeaa ja helppoa käyttää demotarkoituksessa. Demosävellyksen yhteydessä aloin luonnostella erilaisia visualisointeja musiikin funktionalisuudesta. Suunnittelutyötä tehdessäni kävin läpi useita iteraatioita, joista moni tuntui paperilla hyvältä mutta jotka olivat loppujen lopuksi jopa turhan monimutkaisia (ks. liite 3).

Lopullisesta sävellyksestä tuli noin 30 sekuntia pitkä, joka koostui kahdesta osasta. Kehitin sävellykselle instrumentaation, jossa käytin jousi-instrumentteja, kitaraa, kellopeliä ja muutamia perkussioinstrumentteja. Esimieheni Vesa-Matti Mattsson jatkokehitti instrumentaationi pohjalta sovitukset Helsingin Kaupunginorkesterin soittajille, joiden soittoperformansseja meillä olisi mahdollisuus äänittää samassa rakennuksessa sijaitsevassa Tempo Music Studiossa.

Lopullisten instrumenttiversioiden äänitysten jälkeen siirryttiin musiikin miksaukseen. Jotta suunnittelemani adaptiivisen musiikin järjestelmä toimisi, tarvitsisin miksatun musiikin jaoteltuna neljään eri kerrokseen, “stemmiin”. Yksi musiikin kerros sisältäisi aina osan lopullisesta orkestraatiosta. Ensimmäisessä kerroksessa olisi pelkistetysti vain melodiainstrumentti ja sointukierto, toisessa kerroksessa melodian tuplaus ja sointukierto toisella instrumentilla.

Saatuani musiikin kuulostamaan mielestäni hyvältä sekä kokonaisuutena että kerroksiin jaettuna, renderöin kerrokset erillisinä stereotiedostoina ulos Pro Toolsista. Seuraavassa kappaleessa perehdyn musiikkiraitojen käyttöönottoon. Pelimusiikin muuttaminen adaptiiviseksi vaatisi erillisen järjestelmän ohjelmointia.

4.5.2 Adaptiivisen musiikkijärjestelmän ohjelmointi

Luotuani FMOD Studiossa uuden äänitapahtuman pelin taustamusiikille, loin jokaiselle musiikin kerrokselle oman raidan. Lisäsin tapahtuman master-raidalle hienovaraisen kaikuefektin, jotta musiikissa tapahtuvat muutokset kuulostaisivat sulavammilta. Äkkinäisesti musiikista pois putoavat instrumentit saattaisivat kuulostaa erikoisilta ilman minkäänlaista jälkikaiuntaa.

Käytin musiikin elementtien hallinassa hyödyksi aiemmin esittelemääni parametrijärjestelmää. Loin musiikille parametrin “MusicIntensity”, jolla olisi minimi- ja maksimiarvot 0 ja 1. Musiikin saadessa arvon 0 instrumentaatiossa soisi vain sen pohjakerros, sisältäen melodian ja sointukierron. Maksimiarvolla musiikissa soisi koko miksauksen kaikki neljä kerrosta.

Seuraavaksi kohtasin ongelman: minkä pelin funktion asettaisin lähettämään arvoja musiikille? Toisin kuin aiemmin mainitsemassani häviömusiikin esimerkissä, minun ei ollut mahdollista käyttää pelissä olemassa olevaa yksittäistä tapahtumaa kontrolloimaan musiikkia. Adaptiivinen musiikki vaatisi metodin, jolla voisi yhtä lailla lisätä musiikin intensiteettiä kuin laskea sitä.

Päätin rakentaa metodin, joka käyttää hyödykseen pelaajan siirroista kertyviä pistearvoja. Diamond Duossa pelaajan on mahdollisuus saada yhdellä käytetyllä siirrolla vähimmillään 100 pistettä. Maksimissaan taas pisteitä voi tulla niin paljon kuin pelaajan aiheuttama ketjureaktio antaa myöten. Määritin “CalculateMusicIntensity”-metodiin ehtoja eri pistearvoille. Pelaajan saadessa esimerkiksi 100 pistettä annetaan musiikkia kontrolloivalle metodille arvo -0.1. Negatiivisen arvon saadessaan musiikin intesiteetti laskee 10% alaspäin. Päinvastaisessa tilanteessa pelaajan saadessa vaikkapa 500 pistettä, musiikkia kontrolloiva metodi saa arvon 0.1, jolloin musiikin intesiteetti nousee kymmenellä prosentilla.

Tälläinen järjestelmä ei itsessään ota huomioon sellaisia kenttiä, joiden päätarkoitus ei olekaan kerätä pisteitä. Joissain kentissä pisteiden keruun sijaan pitää tuhota peliruudulle ripoteltuja metallilevyjä, tai peliruutua täyttäviä kivenlohkareita. Ratkaisuna ongelmaan käytin hyödyksi jo olemassaolevia äänitapahtumia, joita kutsutaan edellä mainittuja metallilevyjä ja lohkareita tuhotessa. Aina kun peli kutsuu metallin tuhoutumisen äänen, lähetetään musiikkia kontrolloivalle metodille arvo +0.1, jolloin musiikki saadaan jälleen nousemaan intensiteetissä kymmenen prosenttia. Sain adaptiivisuuden toimimaan lopulta tarpeeksi tyydyttävästi. Lisäämällä jälkikaiuntaa master-raidalle sain transitiot eri instrumentaatiotasojen välillä kuulostamaan yllättävän luonnollisilta.

5 Lopuksi

Ennen projektin aloittamista pelinkehitys oli minulle täysin tuntematon prosessi. Elokuvaäänen opiskelun aikana totuin hyviin lineaariseen työskentelymalliin; kuvauksissa äänitetyt raidat otetaan editoitavaksi ja elokuvan ääniraitaa työstetään niin kauan kunnes elokuvan työryhmä on äänisuunnittelijaa myöten tyytyväinen tai kunnes jälkitöille varattu aika loppuu. Pelinkehitystyössä pääsin kokeilemaan hyvin erilaista lähestymistapaa äänisuunnitteluun. Projektin aikana peliin lisätään jatkuvasti uusia ominaisuuksia ja visuaalisia elementtejä, jotka alati muuttavat peliä kokonaisuutena. Joskus ennalta suunnitellut ratkaisut vaikkapa pelimekaniikan suhteen eivät toimikaan käytännössä, jolloin suunnitelmia joudutaan muuttamaan kesken projektin. Haasteena on kyetä dynaamisesti vastaamaan yllättäviin muutoksiin ja olla valmiina tekemään nopeitakin ratkaisuja lyhyiden aikarajojen puitteissa.

Projektin myötä olen oppinyt hyödyntämään pelinkehitykselle ominaisia työkaluja, kuten Unitya ja FMOD Studiota. Molemmissa ohjelmistoissa on omat lainalaisuutensa, joiden oppimisessa kesti oman aikansa. Koen kuitenkin tärkeäksi, että oppimisprosessi tapahtui osana käytännön tekemistä. Olen huomannut oppivani paljon tehokkaammin silloin, kun joudun haastamaan itseäni annettujen tehtävien suorittamiseksi. Tiedän keskittymiskykyni olevan koetuksella, kun joudun lukemaan esimerkiksi pitkiä ohjeistavia tekstejä – yleensä päädyn mielummin sukeltamaan suoraa päätä tuntemattomaan, ja oppimaan yrityksen ja erheen kautta.

Ohjelmistotuntemuksen lisäksi olen saanut kosketuspintaa ohjelmointiin. Ennen Diamond Duon parissa työskentelyä olin kyllä harkinnut useasti suorittavani verkossa JavaScript-ohjelmointikielen peruskurssin, mutta se jäi lopulta vain ajatuksen tasolle. Alkumetreillä olin kauhuissani, ja lähes varma siitä, etten koskaan oppisi ymmärtämään pelissä käytettyä C#-ohjelmointikieltä. Avuliaiden työryhmän ohjelmoijien avustuksella onnistuin kuitenkin selättämään pahimmat pelkoni ohjelmoinnin suhteen. Pian löysinkin itseni rakentelemasta itse omaa adaptiivisen musiikin järjestelmääni. Järjestelmä saattaa olla yksinkertainen, mutta siitä huolimatta olen tyytyväinen lopputulokseen. Asetin itselleni projektin alussa musiikin adaptiivisuuden suhteen päämäärän, jonka pystyin omin avuin saavuttamaan.

Opin myös arvokkaita tietoteknisessä ympäristössä vaadittuja työskentelytaitoja ja käytäntöjä. Diamond Duon yhteydessä pidimme joka viikon aluksi palaverin, jossa suunniteltiin ja käytiin läpi tulevan viikon tehtävät jokaiselle työntekijälle. Viikkopalaverien lisäksi tapasimme joka aamu, jotta kaikilla olisi aina tiedossa mitä kukin on milloinkin tekemässä. Pelinkehityksessä käytetään yleensä jonkinlaista pilvipohjaista versionhallintajärjestelmää, jonka avulla kaikki työryhmän jäsenet voivat tehdä samaan peliin muutoksia ja päivityksiä. Versionhallinnalla myös varmistetaan, että ohjelmistosta on saatavilla toimiva aiempi versio, jos ohjelmistoon tehdään vahingossa vaikkapa virheellinen päivitys.

Diamond Duon parissa työskentely auttoi minua täsmentämään ja kehittämään omaa ammattitaitoani äänisuunnitelijana. Yksi suurimpia ongelmiani luovien töiden parissa on ollut epäjärjestelmällisyys. Minun on paikoin ollut hyvin vaikeaa pitää asioita selkeässä järjestyksessä. Projektin myötä minun oli kuitenkin pakko opetella järjestelmällisyyttä työn tehokkuuden ja mielekkyyden takaamiseksi. Opin järjestelemään Pro Tools-sessiot ja FMOD Studion äänitapahtumat omiin kansiohierarkioihinsa. Selkeällä tiedostojen ja projektien organisoinnila kenen tahansa olisi helppo jatkaa niiden parissa työskentelyä, tilanteen niin vaatiessa.

Ennen kaikkea tunnen poltetta oppia pelien suunnittelusta lisää. Olen kokenut jonkinlaisen uuden heräämisen äänisuunnittelijana oltuani mukana pelinkehitysprosessissa alusta loppuun. Vaikka kyseessä olikin pienemmän skaalan mobiilipeli, projekti on saattanut minut uuden ja äänisuunnittelijalle mielenkiintoisen tarinankerrontaformaatin äärelle. Palan halusta päästä jonain päivänä äänisuunnittelemaan peliä, joka elokuville

ominaisen lineaarisen kerrontatyylin sijasta vie interaktiivisen tarinankerronnan seuraavalle tasolle.

Lähteet

Collins, Karen 2008. Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design. Iso-Britannia: Massachusetts Institute of Technology.

Horowitz, Steve & Looney, Scott 2014. The Essential Guide To Game Audio. USA: Focal Press.

Kolari, Mika 2011. C# Perusteet [verkkodokumentti]
Saatavuus <http://mikakolari.fi/csharp-dotnet/perusteet/luokat/> (luettu 17.4.2015).

Russell, Dave 2012. Video Game Audio: Diegesis Theory [verkkodokumentti]
Saatavuus <http://devmag.org.za/2012/04/19/video-game-audio-diegesis-theory-2/>
(luettu 20.4.2015).

Casual Games Association 2014. Global Market In 2017 [verkkodokumentti]
Saatavuus <http://issuu.com/casualconnect/docs/ccnewzoospringreport-pages?e=233-6319/6014071> (luettu 11.4.2015).

Apple Inc. 2015. Technical specifications [verkkodokumentti]
Saatavuus <https://support.apple.com/specs/> (luettu 26.4.2015).

AppleInsider 2013. Apple ups limit of App Store downloads over cellular to 100MB [verkkodokumentti] Saatavuus <http://appleinsider.com/articles/13/09/18/apple-ups-limit-of-app-store-downloads-over-cellular-to-100mb> (luettu 28.4.2015).

Wikipedia, vapaa tietosanakirja: Olio (ohjelmointi) [verkkodokumentti]. Päivitetty 10.4.2013. Saatavuus <http://fi.wikipedia.org/wiki/Olio_(ohjelmointi)> (luettu 15.4.2015).

Wikipedia, vapaa tietosanakirja: Henkilökohtainen tietokone [verkkodokumentti]. Päivitetty 15.2.2015. Saatavuus <http://fi.wikipedia.org/wiki/Henkilokohtainen_tietokone> (luettu 12.4.2015).

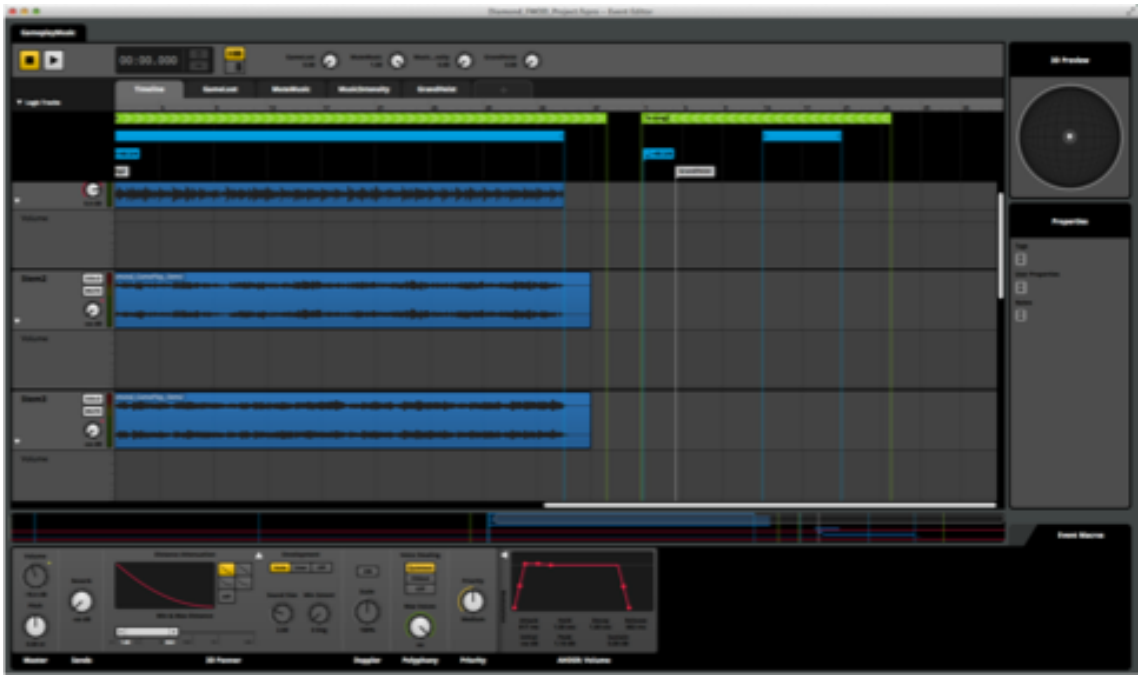
Wikipedia, vapaa tietosanakirja: Tile-matching video game [verkkodokumentti]. Päivitetty 13.2.2015. Saatavuus <http://en.wikipedia.org/wiki/Tilematching_video_game> (luettu 12.4.2015)

Tämän opinnäytetyön teososa ”Diamond Duo” on ladattavissa Suomen App Storesta iOS-laitteille. Saatavuus <https://itunes.apple.com/fi/app/diamond-duo/id963823881>

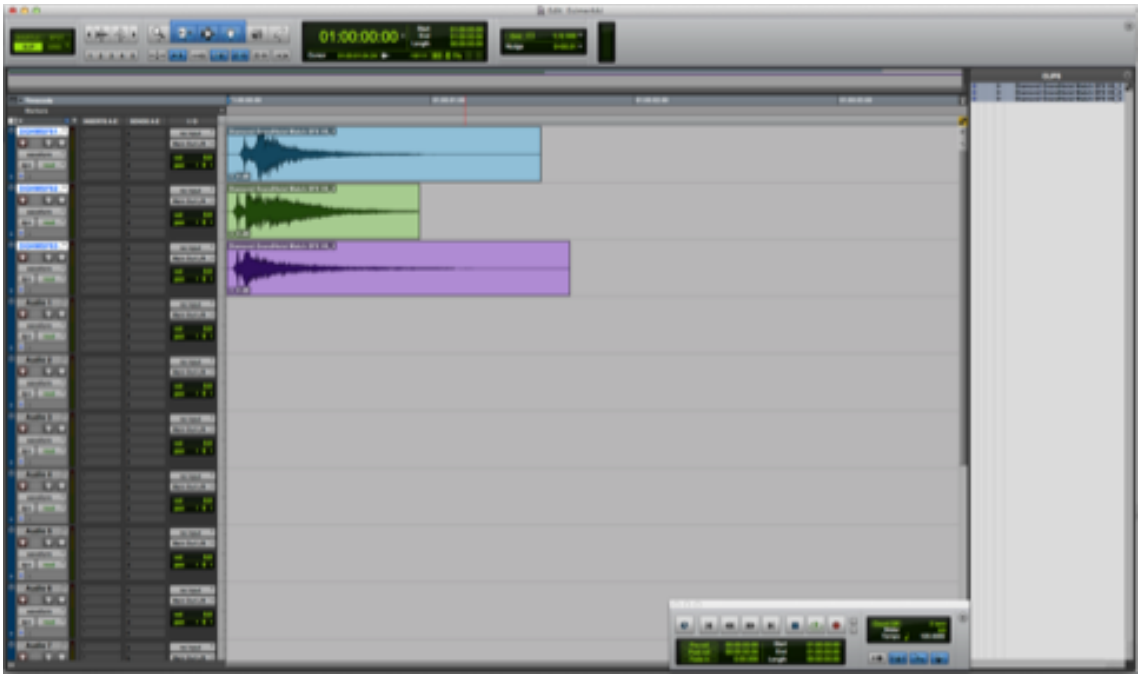
Diamond Duon pelimekaniikan esittely

FMOD Studio- ja Pro Tools -ohjelmistojen käyttöliittymien vertailu

Firelight Technologies FMOD Studio



Avid Pro Tools 11



Eräs suunnitelma adaptiivisen musiikin funktionaalisuudesta